Entorno para el Diseño de Sistemas Basados en Componentes de Tiempo Real

José M. Drake, Julio Luis Medina y Michael Gonzalez Harbour

Grupo de Computadores y Tiempo Real.
E.T.S.I. Industriales y de Telecomunicación.
Universidad de Cantabria
Avda.Los Castros s/n 39005 Santander - España
{ drakej, medinajl, mgh }@unican.es

Resumen. Se presenta una estrategia y un conjunto de herramientas para el diseño de sistemas de tiempo real construidos mediante ensamblado de componentes software. La metodología se basa en diseñar componentes que llevan agregado a su especificación una descripción genérica de su comportamiento temporal, de forma que cuando se construye una aplicación a partir de ellos, se construye a través de herramientas automáticas un modelo de tiempo real de la aplicación, que puede a su vez ser procesado por herramientas de diseño que ayudan a la configuración de los componentes y por herramientas de análisis que permiten garantizar su planificabilidad. El proceso de desarrollo es soportado por una herramienta CASE basada en UML que gestiona las bases de datos con las descripciones de los componentes y los modelos de las plataformas en las que se despliega la aplicación y desde las que se pueden invocar las herramientas de análisis y diseño de tiempo real.

1 Introducción

El objetivo de la tecnología de componentes software es construir aplicaciones complejas mediante ensamblado de módulos que han sido previamente diseñados por otros a fin de ser reusados en múltiples aplicaciones [1]. La ingeniería de programación que sigue esta estrategia de diseño se la conoce por el acrónimo CBSE (Component-Based Software Engineering) y es actualmente una de las tendencias más prometedoras para incrementar la calidad del software, abreviar los tiempos de acceso al mercado y gestionar el continuo incremento de su complejidad. Aunque la tecnología de componentes es ya una realidad en muchos campos, como multimedia, interfaces gráficas, etc., plantea problemas para su aplicación en otros dominios como es el caso de los sistemas de tiempo real. Las dificultades que se presentan, no solo surgen de la mayor diversidad y rigurosidad de los requerimientos que plantea o de las restricciones que se imponen a estos tipos de sistemas, sino principalmente de que se necesitan nuevas infraestructuras y herramientas para implementarla en los entornos de tiempo real [2,3,4,5].

Sin embargo, la necesidad de la tecnología de componentes en el desarrollo de sistemas de tiempo real está siendo manifestada por la industria de automatización, eléctrica, aviación, automóvil, etc. [6,7,8,9]. Empresas líderes en estos campos como ABB han implantado la tecnología de componentes en sus productos, basándola en soluciones y especificaciones propias [6,7], lo cual les ha reportado ventajas en cuanto al mantenimiento y a la gestión de la evolución de sus productos, pero ha supuesto muchos problemas a sus clientes, que encuentran dificultades de compatibilidad de estos productos con aplicaciones desarrolladas por terceros. En otros casos [3,10,11], se ha seguido una solución diferente: se han encapsulado aplicaciones convencionales en módulos que presentan interfaces que ofrecen la conectividad y servicios propios de las tecnologías de componentes estándares mas difundidas (EJB, COM+, CCM, etc.) y se ha buscado conseguir los requerimientos propios de tiempo real, introduciendo en los componentes múltiples parámetros de configuración para que sean establecidos de acuerdo con el entorno en que vayan a operar. Esta estrategia resuelve la integración de los módulos en aplicaciones de mayor nivel a través de ensamblado, pero introduce grandes dificultades en cuanto a poder garantizar la capacidad de mantener las prestaciones no funcionales (fiabilidad, respuesta temporal, etc.) en entornos no bien definidos y que claramente no están concebidos para aplicaciones de tiempo real.

El diseño de componentes de tiempo real reusables presenta dificultades propias respecto del diseño de componentes que no son de tiempo real. En primer lugar, para conseguir las prestaciones de tiempo real se requieren mecanismos de colaboración y sincronización entre los componentes a un nivel mas bajo del que pueden ofrecer las interfaces que utiliza la tecnología de componentes como unidad de especificación e interconexión. En segundo lugar, es habitual que los sistemas de tiempo real sean sistemas embarcados, y en estos, a fin de reducir los costos y hacer posible la integración física, se limitan los recursos disponibles y se emplean entornos heterogéneos que no son los que están previstos en las tecnologías de componentes estándares. Por último, la reusabilidad de los componentes de tiempo real suele estar limitada por la necesidad de ser soportados por sistemas operativos, sistemas de comunicación o bases de datos que ofrezcan servicios específicos de gestión del tiempo, de planificación y de predecibilidad que no son habituales en los entornos de tiempo real.

El diseño de componentes de tiempo real que puedan ser ejecutados en diferentes plataformas HW/SW es una tarea compleja, ya que los componentes presentan diferente comportamiento temporal cuando se ejecutan en diferentes plataformas y en consecuencia, requieren ser verificados o reconfigurados para cada plataforma. Incluso, si se dispone de un catálogo de componentes de tiempo real bien verificados individualmente, habrá que verificar la compatibilidad entre ellos, y comprobar que los mecanismos de comunicación y sincronización retienen las prestaciones temporales previstas en la plataforma que se usa.

La línea de trabajo sobre desarrollo de componentes de tiempo real que está llevando a cabo nuestro grupo se basa en dos antecedentes ya resueltos. En primer lugar, se han desarrollado y se encuentran disponibles núcleos y sistemas operativos

de tiempo real RT_Linux y MarteOS [12,13] que ofrecen sus servicios de acuerdo con el estándar POSIX.13, lo cual proporciona un nivel de referencia estandarizada para la reusabilidad de los componentes en muchas plataformas. En segundo lugar, se han desarrollado métodos y herramientas de diseño y análisis de sistemas de tiempo real (MAST: Modeling and Analysis Suite for Real-Time Applications) [14,15,16] que permiten modelar y analizar sistemas de tiempo real basados en componentes, ya que se basan en el modelado independientemente de las plataformas y de los componentes lógicos y así mismo, ofrece las características de modularidad necesarias para generar los modelos de las aplicaciones en función de los modelos de los componentes.

2 Metodología de diseño

El aspecto central del proceso de diseño de aplicaciones basadas en componentes es la gestión de la funcionalidad a través de las especificaciones de las interfaces y esta es la misma tanto si la aplicación es de tiempo real como si no lo es. Tras una evaluación de algunos de los procesos de diseño basado en componentes descritos en la bibliografía, se ha adoptado el proceso resultante de la extensión del proceso RUP (Rational Unified Process) propuesto por Cheesman [17]. En la figura 1, se muestra un esquema a alto nivel de este proceso. Los bloques representan conjuntos de actividades que dan lugar a resultados tangibles, las flechas gruesas representan su secuenciación y las flechas finas representan el flujo de elementos generados que transfieren información entre ellas. Si comparamos este diagrama con los propuestos en la metodología RUP para la metodología orientada a objetos, se comprueba que las actividades iniciales y finales de definición de requerimientos, prueba y despliegue coinciden, mientras que difieren en que las fases centrales del proceso RUP (análisis, diseño e implementación de objetos) se han reemplazado por otras que representan la especificación, aprovisionamiento y ensamblado de componentes.

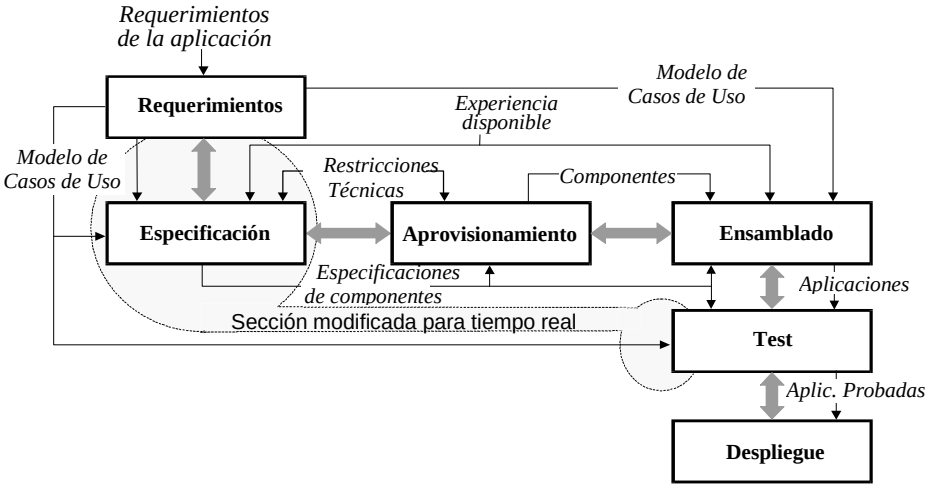

Fig. 1. "Rational Unified Process" modificado para componentes.

Los dos criterios básicos para la elección de este como base de nuestro trabajo son su carácter neutro respecto del proceso de diseño, lo que facilita las extensiones, y su formulación basada en UML que está ampliamente difundido y además lo hace compatible con las herramientas CASE que ya utilizábamos (ROSE de Rational).

La extensión a sistemas de tiempo real que hemos realizado se basa en agregar a las especificaciones de los requerimientos de las aplicaciones y a la descripción de los componentes, nuevas secciones que especifican y describen el comportamiento de tiempo real requerido y ofertado. Así, disponiendo de la descripción de cada uno de los componentes que constituyen una aplicación, junto con un modelo complementario que describe las capacidades de procesamiento de las plataformas hardware/software en que se ejecutan, se puede construir un modelo de tiempo real de la aplicación completa, que es utilizado como base de operación de las herramientas de diseño que ayudan a la configuración óptima de los componentes y de las herramientas de análisis que permiten garantizar la planificabilidad de la aplicación, o en caso negativo mediante análisis de holguras localizar las causas que le impiden satisfacer los requerimientos temporales.

La extensión que se propone solo afecta a tres de los bloques de trabajo del proceso de desarrollo de la figura 1:

- En el bloque de definición de los requerimientos, se ha realizado la extensión necesaria para que se puedan formular los requerimientos de tiempo real de la aplicación. Estos requerimientos se definen de forma independiente para cada modo de operación del sistema, y por cada uno de ellos, se formulan a través de las declaraciones de los conjuntos de transacciones que concurren en él. La declaración de cada transacción de tiempo real incluye la descripción de sus características de disparo, de los conjuntos de actividades que conlleva su ejecución y de los requerimientos de tiempo real que se imponen a lo largo de ella.
- En el bloque de actividades relativas a la especificación de los componentes, se incluyen los procedimientos de descripción del comportamiento temporal del componente. Esto se realiza mediante la especificación de la secuencia de actividades que conlleva la ejecución de cada una de los procedimientos ofrecidos por sus interfaces, y por cada actividad, se describe la cantidad procesado que requiere así como los recursos compartidos que puede requerir y que pueden dar lugar a suspensiones de su ejecución.
- Por último, en el bloque de actividades de test se incluyen las actividades que genera el modelo de tiempo real de la aplicación a partir de los modelos de los componentes que se han utilizado para su ensamblaje y de los modelos de las plataformas hardware/software en que se despliega la aplicación. Así mismo se incluyen, los procesos de análisis del modelo construido, bien para la toma de decisión sobre la configuración óptima de los componentes o bien para verificar su planificabilidad.

La idea básica de la metodología que proponemos es complementar la descripción de los componentes con un modelo de comportamiento temporal detallado, que permita modelar y analizar el comportamiento de tiempo real de las aplicaciones que

hagan uso de ellos. Nuestro método se basa en la capacidad de modelado y de análisis que tenemos, y no en la propuesta de nuevas en estrategias de diseño.

3 Modelo de tiempo real

Los conceptos y recursos de modelado de tiempo real que se proponen proceden de la metodología MAST (Modeling and Analysis Suite for Real-Time Applications) que ha sido desarrollada por nuestro grupo [14,15,16]. Su principal función es simplificar la aplicación de técnicas estándar y bien conocidas de análisis de sistemas de tiempo real, proporcionando al diseñador un conjunto de herramientas para aplicar técnicas de análisis de planificabilidad, de asignación óptima de prioridades o de cálculos de holguras, etc., sin necesidad de que tenga que conocer los detalles algorítmicos de los métodos. Actualmente, MAST cubre sistemas monoprocesadores, sistemas multiprocesadores, y sistemas distribuidos basados sobre diferentes estrategias de planificación, incluyendo planificación expulsora y no expulsora, manejadores de interrupción, servidores esporádicos, escrutinios periódicos, etc.

Características relevantes de la metodología MAST, que la hacen especialmente idónea para modelar sistemas basados en componentes, son:

- Se basa en el modelado independiente de la plataforma (procesadores, redes de comunicación, sistema operativo, drivers, etc.), de los componentes lógicos que se utilizan (requerimientos de procesado, de recursos compartidos, de otros componentes, etc.) y del propio sistema que se construye (partición, despliegue, situaciones de tiempo real, carga de trabajo y requerimientos temporales).
- El modelo se construye con elementos que modelan los aspectos de tiempo real (temporización, concurrencia, sincronización, etc.) de los elementos lógicos (componentes), y en consecuencia da lugar a un modelo que tiene la misma estructura que el código.
- Permite el modelado independiente de cada componente lógico de alto nivel, a través de un modelo de tiempo real genérico, que se instancia en un modelo analizable cuando se completa con el modelo de los componentes de los que depende y se asignan valores a los parámetros definidos en el modelo.

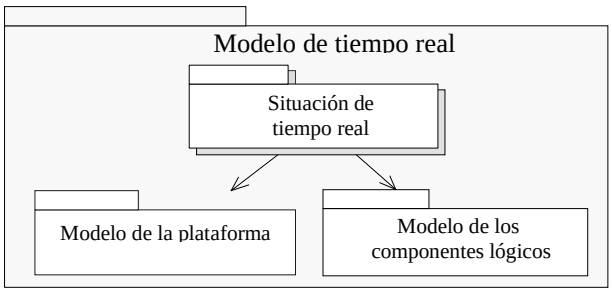

Fig. 2. Secciones del modelo de tiempo real de una aplicación.

- Soporta implícitamente la distribución de componentes, de forma que en función de que un componente se encuentre asignado o no al mismo procesador que otro del que requiere servicios, se incorpora o no el modelo de la comunicación para la invocación del servicio.

El comportamiento de tiempo real de un sistema se descompone en tres secciones independientes pero complementarias que en conjunto describen un modelo que proporciona toda la información estructural y cuantitativa que se necesita para ser procesado por las herramientas de diseño y análisis de que se dispone.

3.1 Modelo de la plataforma

Modela los recursos hardware/software que constituyen la plataforma que ejecuta la aplicación. Modela los procesadores (i.e., la capacidad de procesado, el planificador, el timer del sistema, etc.), las redes de comunicación (i.e., la capacidad de transferencia, el modo de transmisión, los planificadores de mensajes, los drivers de comunicación, etc.) y la configuración (i.e., las conexiones entre los procesadores a través de las redes de comunicación).

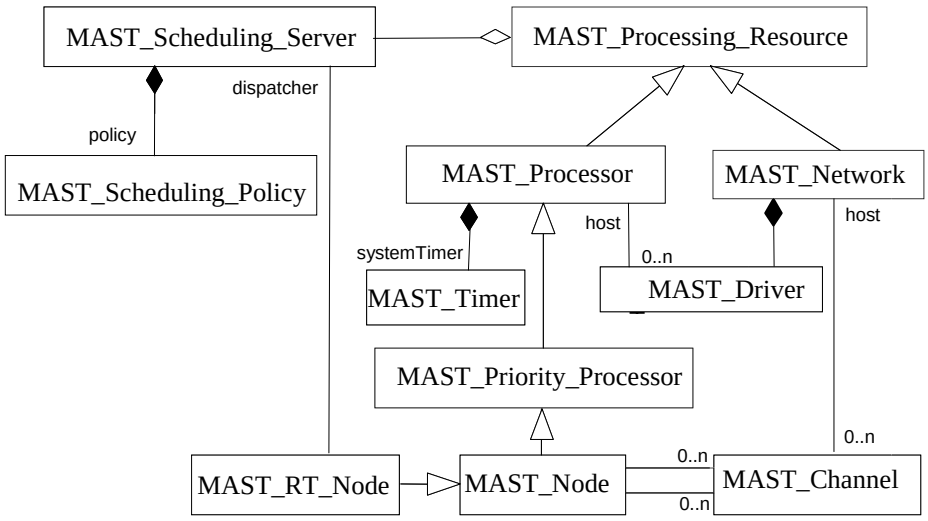

Fig. 3. Principales clases del modelo de la plataforma.

El componente básico del modelo de la plataforma es el Processing_Resource que modela un componente del sistema capaz de ejecutar actividades programadas en él, a través de conjuntos de Scheduling_Server definidos en ellos y a los que se asocian políticas especificas de planificación. En un primer nivel el Processing_Resource se especializa en Processor con capacidad de ejecutar código y Network con capacidad de transferir mensajes. Y en sucesivos niveles se definen recursos más especializados

para los que se definen atributos que describen cuantitativamente su capacidad de procesamiento o de comunicación.

El modelo de la plataforma de un sistema es un diagrama de objetos instanciados que representan los componentes que existen en la plataforma, y la interconexión entre ellos. La naturaleza de los objetos se describe a través del estereotipo, los parámetros del modelo como atributos, y sus valores concretos como valores de inicialización de los atributos.

3.2 Modelo de los componentes lógicos

Modela el comportamiento de tiempo real de los componentes software con los que se construye la aplicación. El modelo de un componente software describe:
- La cantidad de procesado que requiere la ejecución de las operaciones de su interfaz.
- Los componentes lógicos de los que requiere servicios.
- Los procesos o threads que crea para ejecutar sus operaciones.
- Los mecanismos de sincronización que requieren sus operaciones.
- Los parámetros de planificación definidos explícitamente en el código.
- Los estados internos relevantes a efecto de requerimientos de tiempo real.

En la figura 4 se muestran las clases de mas alto nivel que se utilizan para formular el modelo de los módulos software y en particular de los componentes.

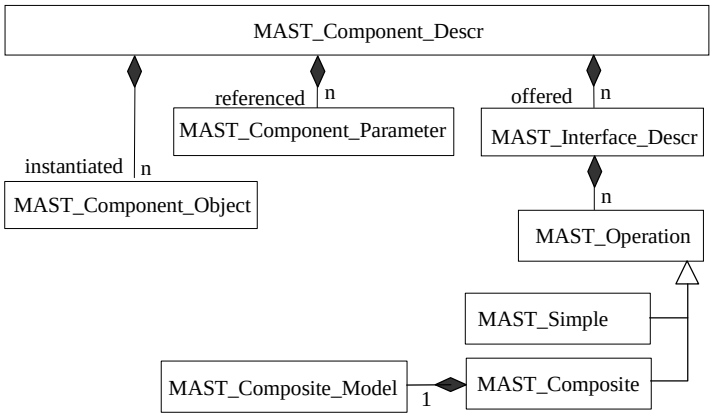

Fig. 4. Principales clases del modelo de tiempo real de los componentes lógicos.

La clase raíz es MAST_Component_Descr. Las instancias de esta clase describen el modelo de un componente. Es básicamente un elemento contenedor de:
- Modelos de operaciones (organizados por interfaces).
- Componentes agregados a él, que serán instanciados por cada instancia del componente.
- Parámetros con los que se configura cada instancia para que represente un modelo específico.

El elemento básico que exporta un MAST_Component es el modelo de temporización de las operaciones que ofrece la interfaz del componente lógico. A diferencia de la descripción lógica, una interfaz no tiene un modelo de tiempo real propio, sino que únicamente constituye un elemento contenedor en el que se organizan los modelos de las operaciones dentro del componente.

Los parámetros declarados en la declaración de un componente constituyen la clave de la capacidad de modelar un tipo de componente mediante un único objeto, que posteriormente a través de sus múltiples instancias sea capaz de describir el comportamiento temporal de componentes bajo entornos muy diferentes. Los parámetros representan diferentes tipos de elementos que son utilizados para describir los modelos de las operaciones. En la descripción del componente los parámetros son solo referencias indefinidas. Sin embargo, en la instanciación del componente le son asignados valores concretos que hacen referencia a objetos instanciados y que completan el modelo de la instancia, y hace posible que diferentes instancias de un mismo componente puedan modelar comportamientos diferentes. Los tipos de elementos que pueden ser referenciados en un componente a través de parámetros, son:

- MAST_Component_Object: Otros componentes de los que depende su funcionalidad.
- MAST_Scheduling_Server: Servidores de planificación que soporten la concurrencia interna de sus actividades.
- MAST_Shared_Resource: Recursos compartidos que son accedidos con exclusión mutua.
- MAST_Operation: Modelo de operaciones.
- MAST_Timing_Requirement: Requerimientos temporales asociados a estados internos
- MAST_External_Event_Source: Modelos de generación de eventos.
- MAST_External_State: Estados internos relevantes de las operaciones.

Los objetos de la clase MAST_Operation modelan la cantidad de procesado y los requerimientos de sincronización de la ejecución de los procedimientos definidos en la interfaz del componente. A alto nivel existen dos clases de operaciones:

- MAST_Simple: Modela la ejecución de una sección de código que no hace referencia a ninguna otra operación. Describe la duración de su ejecución a través de tres parámetros predefinidos: “wcet” worst case execution time, “acet” average case execution time y “bcet” best case execution time. Estos parámetros se describen en tiempo normalizado, que es transformado en tiempo físico cuando se divide por el “Speed_Factor” del recurso de procesamiento en que se ejecuta.
- MAST_Composite: Modela la ejecución de una sección de código que hace referencia (depende) de otras operaciones definidas en la interfaz del mismo componente o de otros componentes. Una operación MAST_Composite requiere ser declarada y también ser descrita. Esto se realiza mediante un diagrama de actividad que se agrega a la propia declaración y que describe las secuencias de actividades que conlleva su ejecución. Así mismo, puede tener definidos

parámetros, lo que particulariza cada invocación de acuerdo con los valores que se le asignan.

3.3 Modelo de las situaciones de tiempo real

Una “Situación de Tiempo Real” representa un modo de operación del sistema junto con un modelo de la carga de trabajo que tiene que ejecutar. Constituye el ámbito en que operará una herramienta de diseño o de análisis. Aunque los modelos de las situaciones de tiempo real son planteados independientemente, comparten el modelo de la plataforma y el modelo de muchos de los componentes software que se utilizan en ellas.

En la figura 5 se muestra los componentes básicos que constituyen el modelo de una situación de tiempo real. Los dos aspectos básicos que describen a una situación de tiempo real son el conjunto de componentes software que se instancian para su ejecución y el conjunto de transacciones que pueden concurrir en ella.

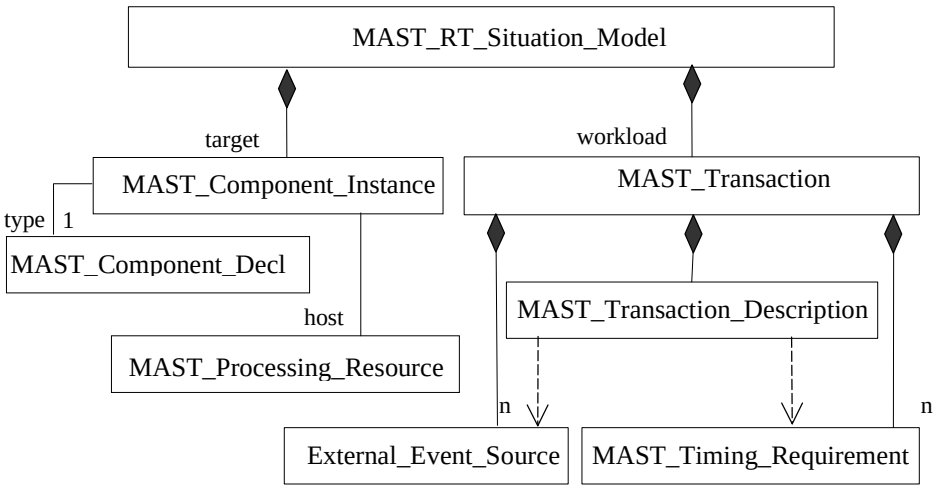

Fig. 5. Principales clases para la descripción de una situación de tiempo real.

En el modelo de una situación de tiempo real se describe la configuración de la aplicación, que consiste en la declaración de cada una de las instancias de componentes software que participan en alguna de las actividades de las transacciones de la situación. La instancia de un componente implica a su vez la instanciación recursiva de todos los componentes que estaban declarados en él. Así mismo, al asignar un identificador a la instancia, se proporciona recursivamente un identificador absoluto (aunque compuesto) a cada componente agregado.

Aunque los objetos de la situación que se modela pueden instanciarse dinámicamente dentro de ella, el modelo de tiempo real es siempre estático, esto es,

todos los objetos que participen han de estar declarados a un mismo nivel y tener asignado un identificador estático.

La configuración de la situación implica también la declaración de su despliegue sobre la plataforma, y a tal fin, a cada instancia de componente software que se declara, se le asigna el procesador en que se ejecuta. Todos los componentes que se instancian por agregación a partir de un componente software, están asignados al procesador a que se asignó este.

La carga de trabajo de una situación de tiempo real se modela como un conjunto de transacciones. Cada Transacción describe una secuencia no iterativa de actividades que se desencadenan como respuesta a un patrón de eventos externos (trigger) que a su vez, sirve de referencia para definir los requerimientos temporales. Cada transacción incluye todas las actividades que se desencadenan como consecuencia del evento de entrada y todas aquellas que requieren sincronización directa con ellas (intercambio de eventos, sincronización por invocación, etc.). No necesitan modelarse dentro de una transacción, otras actividades concurrentes que influyen en su evolución a través de competir por el acceso a recursos comunes, ya sean recursos de procesamiento (Processing_Resource) u objetos protegidos a los que se accede con exclusión mutua. La presencia de estos recursos que son compartidos por las actividades de varias de las transacciones que coexisten concurrentemente dentro de una misma situación de tiempo real y que implican bloqueos en la ejecución de sus actividades, es la causa de que el análisis de tiempo real deba hacerse contemplando simultáneamente todas las transacciones que concurren en la situación de tiempo real.

Las transacciones de tiempo real son componentes básicos de la concepción y especificación de un sistema de tiempo real. Sin embargo, tienen su origen en diferentes fases de su proceso de diseño:

- Transacciones que resultan de la formulación de los requerimientos temporales de las aplicaciones (Event Driven) que se plantean como secuencias de actividades que se desencadenan como respuesta a un evento externo de trigger, junto con las restricciones temporales que se imponen relativas a los instantes en que deben haber concluido estas actividades respecto del trigger que la originó.
- Transacciones que resultan de requerimientos de tiempo real impuestos en la fase de diseño del sistema, como mecanismo interno para satisfacer requerimientos del sistema que en origen pueden no ser de tiempo real.
- Transacciones que se introducen como artificios de análisis para garantizar los requerimientos temporales bajo condiciones de peor caso de sistemas sin eventos externos y con requerimientos temporales entre eventos internos.
- Transacciones sin requerimientos de tiempo real pero que afectan a transacciones con requerimientos de tiempo real que concurren con ellas.

Aunque los cuatro tipos de transacciones tienen origen diferente durante el diseño, convergen al implementarse todas ellas como respuestas a eventos externos generados bien por dispositivos I/O o por el hardware de temporización.

La declaración de una transacción requiere la declaración de tres componentes:

- Trigger_Pattern: Describe la distribución temporal en la generación de la secuencia de eventos externo que la dispara.
- Transaction_Activity: Describe la actividad que se desencadena por cada evento que se produce. Esta actividad se describe mediante un diagrama de actividad agregado a la declaración de la transacción.
- Lista de Timing_Requirements: En cada transacción se definen el conjunto de requerimientos temporales que deben ser satisfechos durante su ejecución.

4 Herramienta de análisis y diseño de tiempo real.

El proceso de desarrollo de aplicaciones basado en componentes que se citó en el apartado 2, requiere de una herramienta que lo soporte. La descripción de cada componente con sus dos vistas, la del que lo usa y la del que lo instala, requiere una información muy extensa que debe ser cotejada para validar la compatibilidad de conexión de los componentes entre sí. En el caso de diseño de sistemas de tiempo real, la herramienta se sitúa en el centro del proceso, ya que el modelo de tiempo real de la aplicación tiene que ser construido a partir de la información de los componentes, y su explotación para las fases de diseño y análisis requiere de la potencia de cálculo para llevarlo a cabo.

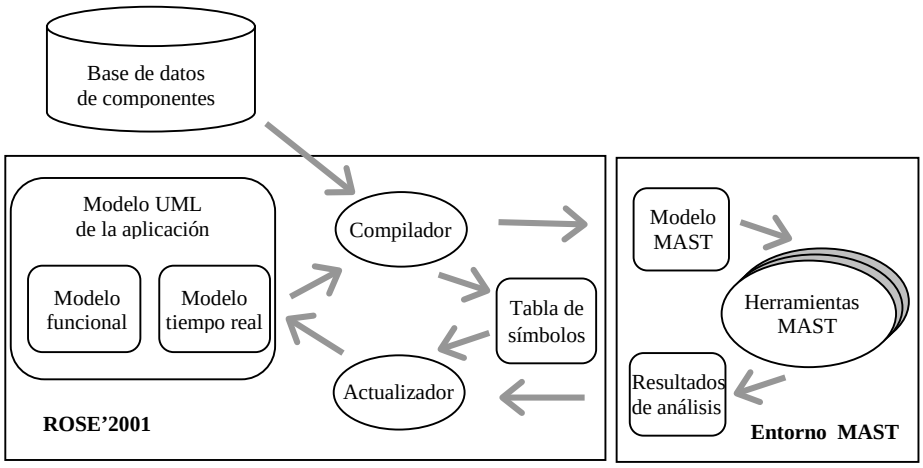

Fig. 6. Conjunto de herramientas para diseño de tiempo real.

Existen tres elementos básicos que constituyen el entorno de instrumentación:

- La herramienta que soporta el proceso de diseño de la aplicación. Está construida a partir de la herramienta ROSE de Rational y su función es servir de soporte de las estructuras UML con las que el diseñador define la aplicación, y las herramientas específicas que generan los modelos de tiempo real y gestionan el acceso a las herramientas externas.

- Las bases de datos de los componentes que contiene las descripciones funcionales y los modelos de tiempo real de los componentes en que se basa el diseño. Estas bases de datos se construyen actualmente con estructuras de ficheros que son a su vez compatibles con los formatos de ROSE.
- Las herramientas de diseño y análisis de tiempo real del entorno MAST, que utilizan ficheros de entrada y salida con formato propio. Entre estas, las que se tienen actualmente desarrolladas (✓) o en desarrollo (-) son:
  - ✓ Holistic analysis
  - ✓ Offset-based analysis
  - - Varying priorities analysis
  - - Multiple event analysis
  - ✓ Monoprocessor priority assignment
  - ✓ Linear HOPA (Holistic optimum priority assignment)
  - ✓ Linear simulated annealing priority assignment
  - - Multiple event priority assignment
  - - Monoprocessor simulation
  - - Distributed simulation

El entorno de diseño que se ha descrito en esta sección es aún parte de un proyecto que se está desarrollando. Muchos de sus componentes, como son la herramienta que soporta la aplicación y las herramientas que implementan los algoritmos de diseño y análisis de tiempo real están ya construidas, pero otros como las bases de datos de los componentes, y los compiladores para la construcción de los modelos de la aplicación aún están en desarrollo.

5 Conclusiones.

Se está realizando la extensión de un entorno de desarrollo de sistemas basados en componentes para que soporte el diseño de sistemas de tiempo real. Esta extensión se basa en la capacidad de modelado del comportamiento temporal de los servicios que ofrecen los componentes y en la generación automática a través de herramientas CASE de modelos de la aplicación, que son procesados a su vez por herramientas de diseño y de análisis de tiempo real.

Con el uso de este entorno, el diseñador puede construir la aplicación por ensamblado de componentes ya elaborados, aunque posiblemente tenga también que diseñar y ensamblar alguno específico que sea propio de la aplicación. Esta fase de diseño está en gran medida conducida por la funcionalidad que se requiere en la aplicación y por la funcionalidad que ofrecen los componentes disponibles. Obviamente si existen requerimientos de tiempo real, en esta fase deben ser también consideradas las capacidades de respuesta temporal de los componentes y seguir en el diseño patrones software que sean adecuados para la planificabilidad de la aplicación. A partir de este punto es cuando se incorporan las capacidades de tiempo real que se han introducido con nuestra extensión. En primer lugar, la herramienta va a proporcionar un modelo global de la aplicación que se genera a partir de los modelos

de comportamiento de los componentes y del modelo de las plataformas hardware/software en que se despliega. En segundo lugar, las herramientas de diseño disponibles ayudan al diseñador en la configuración de la aplicación. Así se puede obtener información sobre las prioridades óptimas de los threads en los procesadores o de los mensajes en las redes, el diseño adecuado de concurrencia y la asignación adecuada de actividades a threads, la asignación adecuada de componentes a los nudos de la plataforma, etc. Por último, las herramientas de análisis podrán garantizar la planificabilidad de la aplicación, si lo es, o identificar los elementos del software que lo impiden.

La principal dificultad que encontramos, para el desarrollo de esta metodología de tiempo real, es disponer de entornos de ejecución de componentes estándares que tengan prestaciones de tiempo real estricto y que sean compatibles con las configuraciones mínimas que son propias de los sistemas embarcados. Las validaciones de la metodología que por ahora hacemos se basan en componentes encapsulados como paquetes Ada y que se ejecutan en plataformas dotadas con sistemas operativos de tiempo real contrastados.

Referencias

1. Szyperski C.: "Component Software: Beyond Object-Oriented Programming" Addison-Wesley, 1999.
2. Isovic D. y Norström C.: "Components in Real-Time Systems". The 8th Int. Conf. On Real-Time Computing Systems and Applications (RTCSA'2002), Tokyo (Japan), 2002.
3. Yau S.S. y Xia B.: "An approach to Distributed Component-based Real-time Application Software Development" Proc. IEEE Int´l Symp.Object-oriented Real-Time Distributed Computing (ISORT'98), April, 1998, pp. 275-283.
4. Lehman M.M. y Ramil J.F.: EPiCS: Evolution Phenomenology in Component-intensive Software" Proposal draft, Dept. of Computing of Imperial College, London, Aug. 2001.
5. Welling A. and Cornwell, P.: "Transaction Integration for reusable hard Real-time Components" Proc. of Ada in Europe, pp. 365 - 378, Springer-Verlag April, 1996.
6. Crnkovic I. Y otros: "A case Study: Demands on Component-based Development" Proc. Of 22nd Int. Conf. Of Software Engineering, Cannes (France), May 2000.
7. Larsson M. y otros: "Development Experiences of a Component-based System" IEEE Proc. of Engineering of Computer Based Systems (ECBS-2000), 2000
8. Norström C. y otros: "Experiences from Introducing State-of-the-art real-time Techniques in the Automotive Industry". Proc. Of 8th IEEE Int. Conf. On Engineering of Computer Based Systems (ECBS01) Washington, April, 2001.
9. Sha L. Rajkumar R. and Gagliardi M.: "Dependable System Upgrade" IEEE Proc. 19th Real-Time Systems Symposium, pp. 440-448, Madrid 1998.
10. Cook J.E. y Dage J.A.: "Highly Reliable Upgrading of Components" Proc. 21st Int. Conf. On Software Engineering. Los Angeles (USA), 1999.
11. Chang E., Annal D. y Grunta F.: "A Large Scale Distributed Object Architecture CORBA&COM for Real Time Systems" IEEE database, 0-7695-0607-0/00., 2000.
12. Aldea M. y González Harbour M.: "MaRTE OS: An Ada Kernel for Real-Time Embedded Applications." International Conference on Reliable Software Technologies, Ada-Europe-2001, Leuven, Belgium, LNCS, May 2001.

13. Aldea M. y González Harbour M.: “POSIX-Compatible Application-Defined Scheduling in MaRTE OS”. Proceedings of 13th Euromicro Conference on Real-Time Systems (WiP), Delft, The Netherlands, June 2001.
14. González Harbour M., Gutiérrez J.J, Palencia J.C. and Drake J.M.: “MAST: Modeling and Analysis Suite for Real-Time Applications” Proceedings of the Euromicro Conference on Real-Time Systems, June 2001.
15. Medina J.L.,González Harbour M.,. Drake J.M:” ”MAST Real-time View: A Graphic UML Tool for Modeling Object_Oriented Real_Time Systems”, RTSS, 2001, December, 2001.
16. Medina J.L., Gutierrez J.J., González Harbour M., Drake J.M.: “Modeling and Schedulability Analysis of Hard Real-Time Distributed Systems based on ADA Componentes.” Ada Europe, 2002
17. Cheesman J. and Daniels J.: “UML Components: A simple Process for Specifying Component-Based Software”, Addison-Wesley,2000.
18. D’Souza D.F. and Wills A.C.: “Objects, Components and Frame-works in UML: the Catalysis approach”. Addison-Wesley, 1998.
19. Gomaa H.: “Designing Concurrent, Distributed and Real-Time Application with UML”. Addison-Wesley, 2000.